ITS Technical Bulletin 239
DB2 VERSION 4 MIGRATION

Issued Date: 22 Apr 1996
Effective Date: 06 May 1996
Section/Groups: Database/CICS Administration
Submitted By: Ted Misiak
Approved By: Sandy Neal

This technical bulletin outlines a schedule for the migration of DB2 Version 4.

- We plan to install DB2 Version 4.1 on Saturday, May 4th 1996.

The installation process should take about 6 hours, during which DB2 Development (CPU4) will be down. All CICS regions on CPU4 which connect to DB2 will be brought down. When the new version of DB2 comes up, we will rebind all plans and packages in order to take advantage of the new/improved DB2 optimizer.

- We will start to convert DB2 indexes to Type 2 indexes one week after the implementation date.

This will involve altering the existing indexes using the SQL ALTER statement followed by running the RECOVER index utility. All DB2 applications will benefit from Index Type 2. This is a requirement for row-level locking.

- The decision to go to row-level locking will involve a careful analysis of each application, since this type of DB2 lock introduces trade-off issues (they are not always desirable). Therefore, the implementation of row-level locking will be an on-going process for quite sometime.

- We are in the process of testing the Data Sharing feature of DB2 Version 4 on CPU0 and CPU1.
  It has not been determined yet when this will be implemented in Development (CPU4).

There are many other new features in DB2 Version 4. The following are brief explanations of these new enhancements:

Row Level Locking - Up to now, when an application processed a row, it made all other rows stored on the same page unavailable to other users. This has changed. The smallest unit of lock could be a row, as opposed to a page. This should improve concurrency and reduce contention, but potentially many more locks may be necessary using row locking; if an application processes many rows in a sequential fashion, row level locking may be very undesirable because obtaining each lock incurs a cost to the CPU. This is why Page Locks have not been discontinued and

remain an option to be considered.

Type 2 Index - when an index is used to access data, DB2 no longer puts a lock on the Index page where the pointer (RID) resides; this allows for much greater concurrency because waiting on Index Locks to be released in order to access data pages has been eliminated. Many concurrency problems were attributable to the contention on Index pages, not on Data pages. This bottleneck has been eliminated in Version 4.

Also, when a row is deleted, its corresponding key in the Index is 'pseudo deleted' - not physically deleted. This improves response time, because 'pseudo delete' just turns on a bit in the RID flag, and does not require the reorganization of the page.

Uncommitted Read (UR) - in addition to the elimination of Index Locks, it is possible to ask DB2 to disregard locks that other applications might be holding. For example, if you would like to compute employees' average salary, but another application is updating the table and denying you an access, now DB2 will let you proceed with your request. Your SQL statement must be SELECT and the UR (Uncommitted Read) clause is specified with SELECT as a Bind parameter.

Data Sharing - allows up to 32 different DB2's (subsystems) to have Read/Write access to the same, shared DB2 data. Benefits:

(1) In the event one DB2 should fail, another DB2 will take over, improving availability.

(2) Ability to spread over/balance workload among many DB2's, improving response time. The reason two or more DB2 systems can work with the same DB2 data (tables) is that these DB2 systems share the same DB2 Catalog and Directory. All DB2 knows is what it reads in it's Catalog and Directory, so by reading the same thing they can be aware of each others DB2 objects (data). The activities of the different DB2 systems are coordinated through the Coupling Facility. If more than one DB2 system is interested in the same data, Global Buffer Pool, which exists in the CF (Coupling Facility), will be used to bring (cache) the page. Information about which DB2 system is presently updating DB2 data (locking), is kept in the Coupling Facility; and lastly, the status of each DB2 data base is stored in the System Communication Area which also resides in the Coupling Facility. Just because DB2 data exists in the DB2 system which participates in Data Sharing, does not mean that this data will be accessible to all DB2 systems. It is possible to 'localize' DB2 tables by defining them in a Buffer Pool that does not have a corresponding Global Buffer Pool.

Query CPU Parallelism - huge DB2 tables are often split into partitions.
I/O Parallelism, which has been implemented in previous version, allows I/O operations to overlap with one another; in other words - prefetching of data from different partitions is concurrent. However, the CPU processing is still sequential.

CPU Parallelism, introduced in Version 4, allows concurrent CPU processing against different partitions to take place. It is done by scheduling multiple SRB's to process a single query. It

works only against partitioned table spaces and requires MVS 5.2.

Outer Join - since the beginning of DB2, join was defined as combining of data from one or more tables based on matching columns; that is to say, the rows from two different tables were combined if, and only if, they had the same (matching) values in one or more of their columns.

Outer Join, however, allows unmatched rows from one or both tables to be included in the result of the join. So the concept of join no longer implies that tables must have common values (data).

SQL - when defining a DB2 table it is possible now to indicate what valid values a column can contain. This should eliminate the need to code validity checking routines in application programs. Also, during the table definition, you could add a clause to restrict dropping the table, preventing accidental loss of data.

Utilities - Copy utility supports an option to allow the copy to be done via DFSMS Concurrent Copy. The benefit of Concurrent Copy is that it appears to be an instantaneous process, without restricting the access to the data. It requires MVS 4.3, DFSMS 1.1, 3990-3 or 3990-6 storage controller.

Recover utility will use DFSMS if image copies are done thru DFSMS Concurrent Copy. It appears that DB2 is trying to get out of the business of copying/restoring files, and concentrate on what it can do best, like guaranteeing data integrity, powerful support for retrieval of data (joins, unions), etc.

Archive Log - DB2 maintains two copies of Active Log, and it is possible now to archive first and second copy of the Active Log to different device types. One copy could be archived to DASD to facilitate fast recovery, while the second Archive Log copy going to the tape might be sent offsite.

Secondary Allocation - the old rule for default secondary allocation (3 pages) caused many repeated requests; in Version 4 the larger of: 10% of PRIQTY (primary allocation) or 3 pages is used.

Down-Level Detection - it is possible to replace DB2 datasets by mistake with an outdated copy. This may not be detected immediately, and applications may run against down-level data. In Version 4, DB2 maintains Current Level_ID for each Data Base, and it will issue a message if it finds a down-level data set.